Writing XML

Creating Well Formed Markup

Well Formed Markup

- XML Documents can be complex or simple
  - A complex document may:
    - Mix multiple XML languages
    - Reference one or more XML Schemas
    - Contain several different kinds of links
    - Be associated with XSLT stylesheets
  - The simplest document is self contained
    - It contains arbitrary markup
    - It does not link to external resources
- Every XML Document must be well-formed
  - It must adhere to the basic syntax rules of XML
  - Otherwise the XML parser will be unable to process it

Well Formed Markup

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<presentation>
    <slide title = “Intro to XML”>
        <summary>This slide introduces basic concepts</summary>
        <para>The first paragraph</para>
        <graphic type=“bmp” source=“./images/pic1.bmp”>
            <caption>How XML is created</caption>
        </graphic>
    </slide>
</presentation>
```

The XML Identifier

- An XML file should begin with the XML declaration
  - Such as <?xml version="1.0" encoding="UTF-8"?>
  - This declaration tells the parser which version of XML is in use and what character encoding to expect
    - Character encodings are a significant impediment to portability
    - XML 1.1 (aka Blueberry) has been proposed to fix some of these
  - The document can also be declared to be 'standalone'
    - <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
- The XML declaration is optional
  - But its always a good idea to put it in
  - MSXML treats it as a processing instruction

Tags and Elements

- XML is mostly made up of tags enclosing content
  - There are three types of tag:
    - Start tags, such as <paragraph>
    - End tags, such as </paragraph>
    - Empty tags, such as <paragraph/>
- Elements are made up of a start tag, an end tag and any content
  - An empty tag is an element with no content
- An element has one of four Content Models:
  - Element only (only other elements appear)
  - Text only (only normal content appears)
  - Mixed (both elements and content are present)
  - Empty (the element contains no content)

Content Models

Element Only

```
<paragraph>
    <text>ABC</text>
    <text>DEF</text>
    <text>GHI</text>
</paragraph>
```

Text Only

```
<paragraph>
    ABC
    DEF
    GHI
</paragraph>
```

Mixed

```
<paragraph>
    ABC
    <text>DEF</text>
    GHI
</paragraph>
```

Empty

```
<paragraph></paragraph>
```

```
<paragraph/>
```

Elements

- All content in the doc occurs inside a single element
  - Just as an HTML page occurs inside <html>…</html>
  - This is referred to as the Document Element
- Other elements are nested inside this document element
  - This enables the XML Parser to build a tree structure of the document in memory
- We use genealogical terminology to locate elements
  - An element may have other elements as children, descendants, ancestors, siblings etc…
  - The document element is the only element without a parent

Proper Use of Elements

```
<?xml version="1.0" encoding="UTF-8">
<doc>
    <a/>
    <b></b>
    <c>Some text</c>
</doc>
```

```
<?xml version="1.0" encoding="UTF-8">
<doc>
    <a/>
    <b></b>
</doc>
<c>Some text</c>
```

```
<?xml version="1.0" encoding="UTF-8">
<doc>
    <a>
        <b>
            Some Text
            <c/>
        </b>
    </a>
</doc>
```

```
<?xml version="1.0" encoding="UTF-8">
<doc>
    <a>
        <b>
            <c>
        </b>
            </c>
    </a>
</doc>
```

Attributes

- Attributes are properties of an element
  - They are written as name/value pairs declared inside the start tag
  - An element can have any number of attributes
    - But two attributes cannot have the same name
      - Unless they belong to different namespaces (see later)
  - Attribute values must be in single or double quotes
    - One type of quote can be nested inside another
  - Attribute values cannot contain ‘<‘ or ‘&’
    - These must be escaped to avoid confusing the parser
- Attributes do the same job as nested elements
  - They are superfluous but useful to define certain characteristics
  - Attributes should be used where a piece of content is:
    - Inherent to the element you are defining
    - Takes a limited number of values

Attributes

```
<customer>
    <type>Retail</type>
    <title>Mr</title>
    <forename>Dave</forename>
    <surname>Jones</surname>
    <company>
        <title>Widgets Are Us</title>
        <location>Belfast</location>
        <vat-no>154 327 859</vat-no>
    <company>
</customer>
```

```
<customer type="retail" title="Mr">
    <forename>Dave</forename>
    <surname>Jones</surname>
    <company vat-no="154 327 859">
        <title>Widgets Are Us</title>
        <location>Belfast</location>
    <company>
</customer>
```

Processing Instructions

- PI's allow a document to signal the parser
  - Similar to the way pragmas work in C/C++
  - If the parser cant understand a PI it is ignored
- PI's are written as '<?target instructions ?>'
  - The target is the name of the PI
    - PI's cannot have names starting with 'xml'
  - The instructions are an arbitrary sequence of chars
    - Often the instructions are formatted as attributes
  - The PI commonly used for stylesheets is:
    - <?xml-stylesheet href="stylesheet.xsl" type="text/xsl"?>
- PI's can occur anywhere in the XML Document
  - But your parser may only look for them in certain places…

Comments

- XML comments work the same as in HTML
  - The delimiters are ‘<!--’ and ‘-->’
  - Comments can stretch over many lines
- Comments can occur anywhere in the document
  - They do not count as content but are not stripped out by the XML parser
  - All characters between the delimiters are held inside a comment node in the tree
  - This could then be read and used by:
    - An XSLT stylesheet
    - Code using an API such as DOM

Whitespace

- Whitespace is effectively ignored in XML documents
  - Multiple spaces or new lines are treated as a single space character
  - It does not matter to the parser whether your document is ‘nicely’ formatted or a continuous string
  - All editors reformat XML files to make them more legible
- Some XML languages define ‘whitespace sensitive’ elements
  - In the same way as <PRE> </PRE> tags in HTML
  - The **xml:space** attribute forces whitespace to be preserved

```
<address xml:space="preserve">
    <!-- text goes here -->
</address>
```

Entities

- An entity is like a hard-coded variable
  - It will be textually replaced with its value when it is referenced
  - This is the simplest way to use reserved characters in your markup
- Entities can be used for
  - Illegal characters
  - Unicode symbols
  - Hard coded strings

```
&amp;      equivalent to &#38;
&lt;       equivalent to &#60;
&gt;       equivalent to &#62;
&apos;     equivalent to &#39;
&quot;     equivalent to &#34;
&copyright_message;
&internal_use_only;
```

CDATA Sections

- Entities are impractical where the markup must contain blocks of equations or source code
  - The ‘<![CDATA[‘ and ‘]]>’ delimiters can be used instead
- CDATA sections contain content that would otherwise be recognized as markup
  - For example SQL queries in J2EE configuration files
  - CDATA sections cannot be used to hold binary data

```
<![CDATA [
        <step1>Install XMLSpy into "/home/xmlspy".
        <step2>Create project.
        <step3>Add project files.
]]>
```

Tips on Writing XML

- Choose a convention and be consistent
  - For example ‘thisIsAnElementName’
- Use attributes to increase clarity
  - Especially for properties with fixed values
- Be verbose when naming collections of elements
  - ‘CustomerList’ is a better choice than ‘Customers’

```
<Customers>
    <Customer id=“no1”/>
    <Customer id=“no2”/>
</Customers>
```

```
<CustomerList>
    <Customer id=“no1”/>
    <Customer id=“no2”/>
</CustomerList>
```

Examples of XML Languages

Rich Site Summary

- RSS is a language for writing change notifications
  - Designed to let portals to display news headlines
- Applications syndicate changes via an RSS ‘feed’
  - Clients subscribe to the feed and regularly receive descriptions of new content
  - Each new item has a title, a URL and a description
- RSS is becoming widely used across the WWW
  - Seven different versions of the standard exist
  - RSS could be used to advertise any type of change

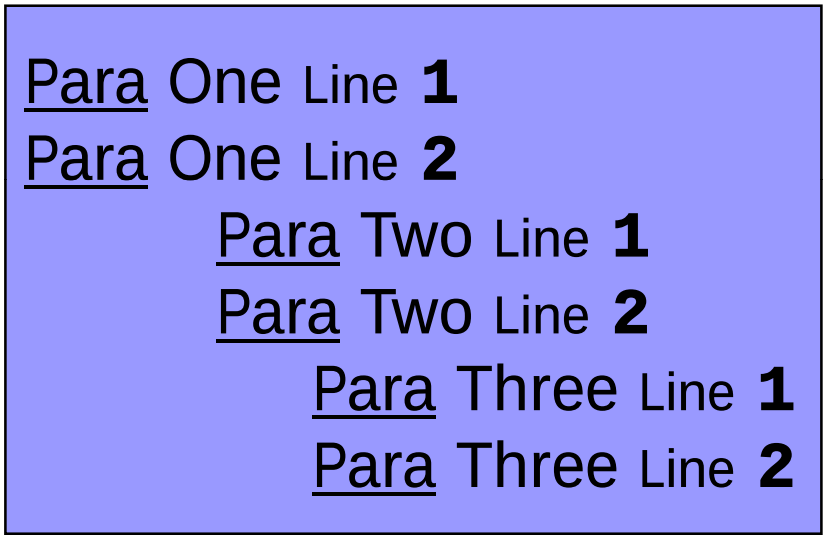
RSS Example

```
<rss version="0.91">
  <channel>
    <title>CMM</title>
    <link>http://www.CMM.com/</link>
    <description>Up to date news from around the world</description>
    <language>en-us</language>
    <item>
      <title>UK Joins Euro</title>
      <link>http://www.cmm.com/pub/story1.html</link>
      <description>UK to join Euro Zone in 2004</description>
    </item>
    <item>
      <title>Raindrops Keep Falling</title>
      <link>http://www.cmm.com/pub/story2.html</link>
      <description> The wettest summer for 250 years </description>
    </item>
   </channel>
</rss>
```

Wireless Markup Language

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN“
          "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
   <card id="no1">
      <p align="left">
         <u>Para</u> <b>One</b> <small>line</small><strong>1</strong><br/>
         <u>Para</u> <b>One</b> <small>line</small><strong>2</strong>
      </p>
      <p align="center">
         <u>Para</u> <big>Two</big> <small>line</small><strong>1</strong><b/>
         <u>Para</u> <big>Two</big> <small>line</small><strong>2</strong>
      </p>
      <p align="right">
         <u>Para</u> <em>Three</em> <small>line</small><strong>1</strong>
         <u>Para</u> <em>Three</em> <small>line</small><strong>2</strong>
      </p>
   </card>
</wml>
```

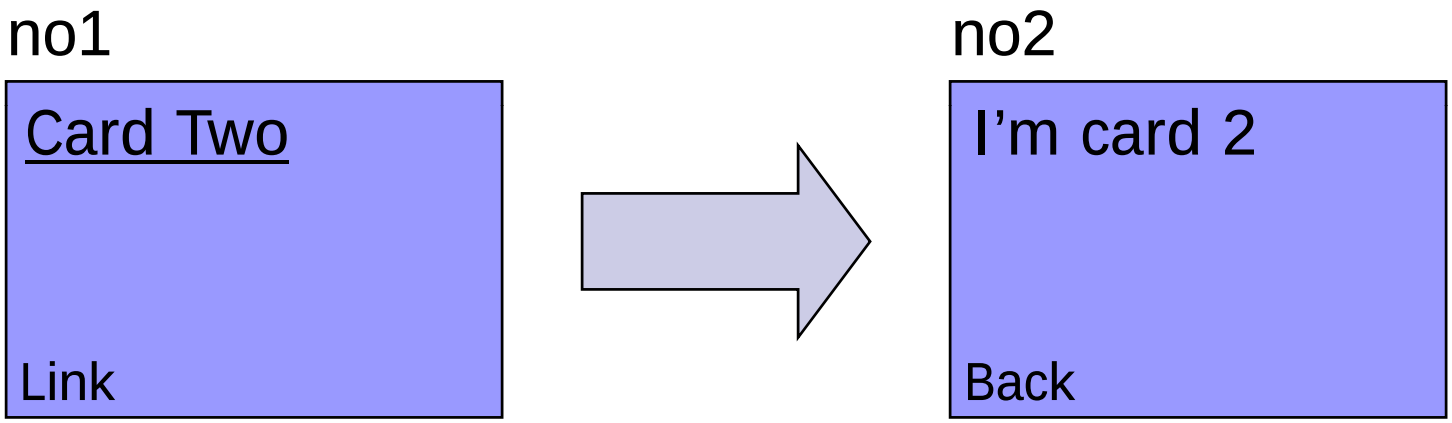
Wireless Markup Language



Wireless Markup Language

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
 "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
	<card id="no1">
		<p>
			<anchor>
				<go href="#no2"/>
				Card Two
			</anchor>
		</p>
	</card>
	<card id="no2">
		<p>I'm card 2</p>
	</card>
</wml>
```

Wireless Markup Language



Scalable Vector Graphics

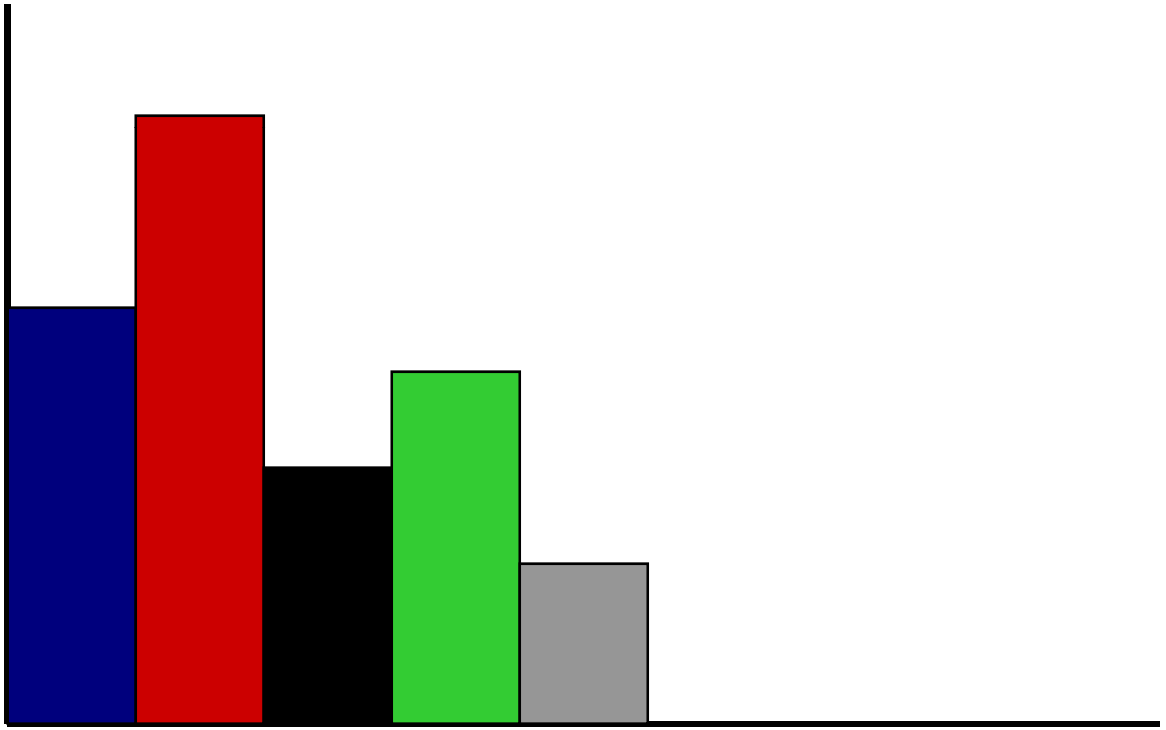
```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-stylable.dtd">

<svg xml:space="preserve" width="5.0in" height="5.0in" viewBox="0 0 200 200">

        <text style="fill:red;" y="15">This is SVG.</text>
        <path style="stroke:black;" d="M 50 50 L 50 190 z"/>
        <path style="stroke:black;" d="M 50 190 L 250 190 z"/>
        <rect x="50" y="190" style="fill:blue;" width="15" height="-70"/>
        <rect x="65" y="190" style="fill:red;" width="15" height="-100"/>
        <rect x="80" y="190" style="fill:black;" width="15" height="-40"/>
        <rect x="95" y="190" style="fill:green;" width="15" height="-60"/>
        <rect x="110" y="190" style="fill:grey;" width="15" height="-20"/>
</svg>
```

SVG Output

This is SVG



XSL Formatting Objects

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
      <fo:layout-master-set>
            <fo:simple-page-master master-name="simple“ page-height="29.7cm" page-width="21cm"
                  margin-top="1cm" margin-bottom="2cm" margin-left="2.5cm" margin-right="2.5cm">
            <fo:region-body margin-top="3cm"/>
            <fo:region-before extent="3cm"/>
            <fo:region-after extent="1.5cm"/>
            </fo:simple-page-master>
            </fo:layout-master-set>

      <fo:page-sequence master-reference="simple">

            <fo:flow flow-name="xsl-region-body">
                  <fo:block font-size="18pt" font-family="sans-serif"
                        line-height="24pt“ space-after.optimum="15pt background-color="blue“
                              color="white“ text-align="center“ padding-top="3pt">

                        Extensible Markup Language (XML) 1.0
                  </fo:block>
```

XSL Formatting Objects

```
                <fo:block font-size="12pt" font-family="sans-serif" line-height="15pt“
                        space-after.optimum="3pt“ text-align="justify">
                        The Extensible Markup Language (XML) is a subset of SGML that is
                        completely described in this document. Its goal is to enable generic SGML
                        to be served, received, and processed on the Web in the way that is now
                        possible with HTML. XML has been designed for ease of implementation
                        and for interoperability with both SGML and HTML.
                </fo:block>
               <fo:block font-size="12pt" font-family="sans-serif" line-height="15pt"
                        space-after.optimum="3pt“ text-align="justify">
                        The Extensible Markup Language (XML) is a subset of SGML that is
                        completely described in this document. Its goal is to enable generic SGML
                        to be served, received, and processed on the Web in the way that is now
                        possible with HTML. XML has been designed for ease of implementation
                        and for interoperability with both SGML and HTML.
                </fo:block>
            </fo:flow>
        </fo:page-sequence>
</fo:root>
```

XSL Formatting Objects

Extensible Markup Language (XML) 1.0

The Extensible Markup Language (XML) is a subset of SGML that is completely described in this document. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

The Extensible Markup Language (XML) is a subset of SGML that is completely described in this document. Its goal is to enable generic SGML to be served, received, and processed on the Web in the way that is now possible with HTML. XML has been designed for ease of implementation and for interoperability with both SGML and HTML.

XQuery

- XQuery is the XML version of SQL
  - It builds on XPath to add querying functionality
  - Note that XQuery is read-only at present
- Databases become collections of XML documents
  - The underlying storage format is unimportant
- This shortens the path from retrieval to display
  - Especially for simple Use Cases
- XQuery will soon be universally supported
  - Native XML databases support it already
  - It is one of the major features in SQL Server 2005

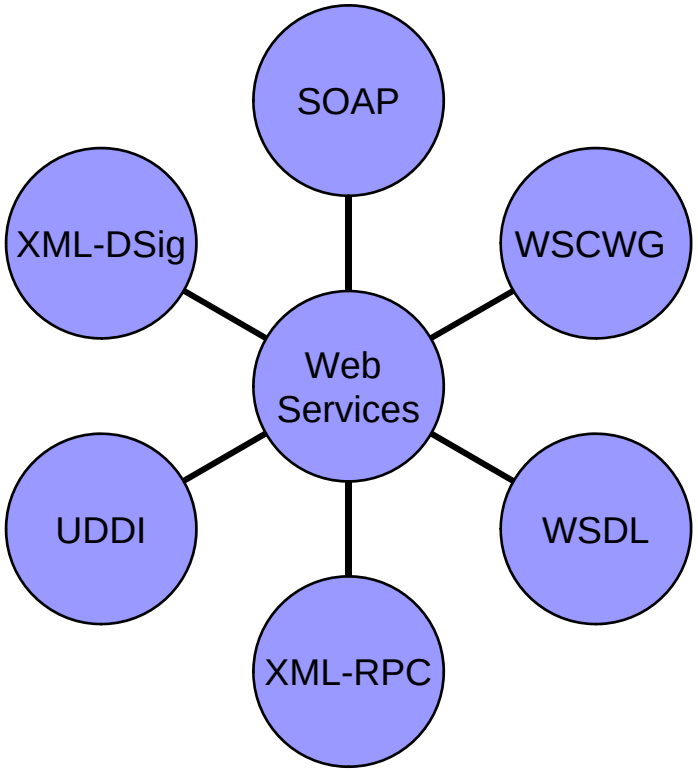
XQuery

```
<table rules="all">
    <tr><th colspan="3">Speeches Over Twelve Lines</th></tr>
    <tr><td>Speech No</td><td>Speaker</td><td>Line Count</td></tr>
    {
        for $speech at $position in doc("muchAdoAboutNothing.xml")//SPEECH
        where count($speech/LINE) > 12
        return
            <tr>
                <td>{ $position}</td>
                <td>{ $speech/SPEAKER/text() }</td>
                <td>{ count($speech/LINE) }</td>
            </tr>
    }
</table>
```

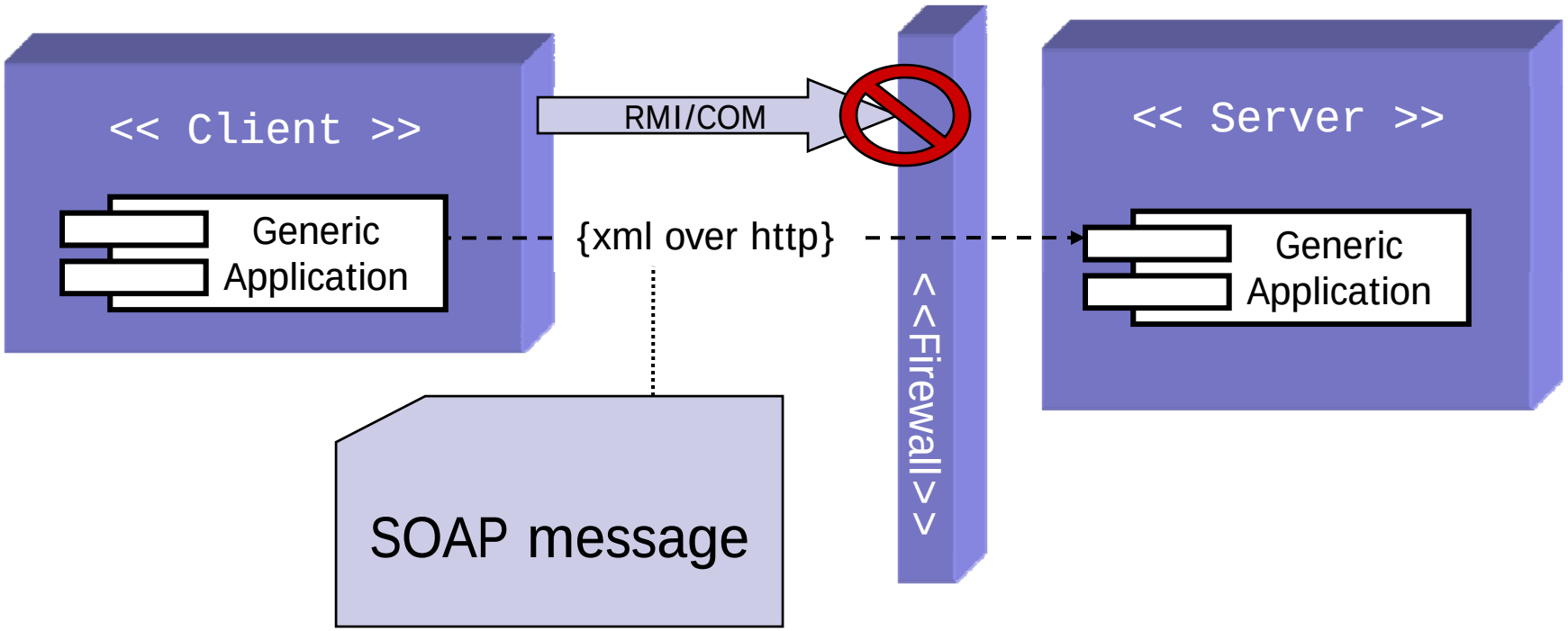
Web Services Vs Semantic Web

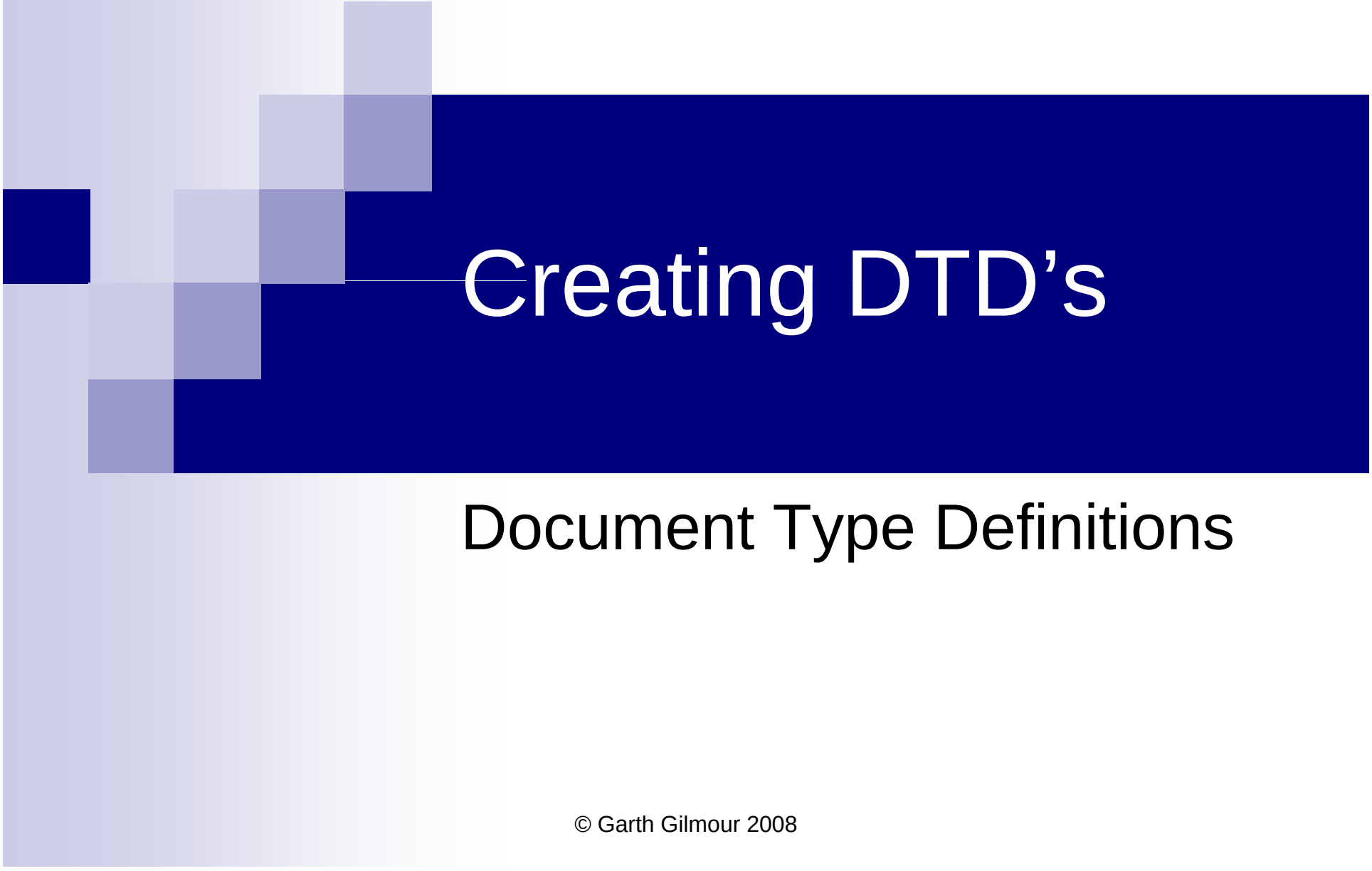
- XML enables us to create a Web where:
  - Many different languages co-exist
  - Data is king and presentation is layered on
  - The browser ‘knows’ what the data means
  - The user can be removed from B2B e-commerce
- Two visions exist for the future of the web
  - The Web Services vision (largely commercial)
  - The Semantic Web vision (largely academic)
- These two perspectives are not mutually exclusive
  - But there may not be enough space for both…

Web Services



SOAP





Document Type Definitions

Document Type Definitions

- Standalone documents are of limited use
  - They rely on the authors knowledge of structure
  - There is no way to guide multiple authors on the correct structure of a document
- What is required is a schema for each XML based language that is created
  - An XML document is an instance of an XML schema just as a database implements its relational schema
- There are two ways to write an XML schema
  - The Document Type Definition standard
  - The newer XML Schema standard

Document Type Definitions

- DTD's are the SGML way to define a document
  - They use a separate syntax to XML documents
  - Support for specifying data types is limited
- Originally DTD's were all that was available
  - XML Schema was not released until 2001
  - Microsoft developed its own XDR Schema standard in parallel with the W3C XML Schema (they are not competitors)
- DTD's are still in use for pre 2002 standards
  - Since then J2EE and .Net have switched to schemas
  - Older Microsoft products support DTD's and/or XDR Schema
  - DTD's are still useful for simple XML applications and as a stepping stone to learning XML Schema

Creating DTD's

- A DTD is made up of rules
  - ELEMENT rules define the name and content model of an element
  - ATTLIST rules define a set of attributes associated with an element
  - ENTITY rules define entities
- DTD's can be loaded from two places
  - Internal DTD's are supplied with the XML document
    - This makes the document self contained but limited
    - The DTD is delimited by the '<!DOCTYPE name [' and ']>' tags
  - Enternal DTD's are referenced from the XML document
    - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN">
    - <!DOCTYPE porder SYSTEM "purchaseOrder.dtd">

Example External DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE course SYSTEM "course.dtd">
<course title="XML For Beginners">
    <prerequisite>Maths</prerequisite>
    <moduleList>
        <module index="1" title="Introduction"/>
        <module index="2" title="Fundamentals"/>
        <module index="3" title="Conclusions"/>
    </moduleList>
</course>
```

Example External DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT course (prerequisite, moduleList)>
<!ELEMENT prerequisite ANY>
<!ATTLIST course
        title CDATA #REQUIRED
>
<!ELEMENT module EMPTY>
<!ATTLIST module
        index (1 | 2 | 3) #REQUIRED
        title (Conclusions | Fundamentals | Introduction) #REQUIRED
>
<!ELEMENT moduleList (module+)>
```

Example Internal DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE course [
        <!ELEMENT course (prerequisite, moduleList)>
        <!ELEMENT prerequisite ANY>
        <!ATTLIST course title CDATA #REQUIRED>
        <!ELEMENT module EMPTY>
        <!ATTLIST module  index (1 | 2 | 3) #REQUIRED
                title (Conclusions | Fundamentals | Introduction) #REQUIRED>
        <!ELEMENT moduleList (module+)>
]>
<course title="XML For Beginners">
        <prerequisite>Maths</prerequisite>
        <moduleList>
                <module index="1" title="Introduction"/>
                <module index="2" title="Fundamentals"/>
                <module index="3" title="Conclusions"/>
        </moduleList>
</course>
```

The Element Rule

- The <!ELEMENT> defines an XML element:
  - <!ELEMENT element_name element_type>
  - The type of the element can be
    - (#PCDATA) (parsed character data/text)
    - ANY (any element type)
    - Other elements as contents
      - Grouped using BNF
      - Specific order (E1,E2?,E3)
      - Optional content (E1|E2)*

Defining Content Models

Empty content	<!ELEMENT customer EMPTY>
Text content	<!ELEMENT customer (#PCDATA)>
Well-formed content	<!ELEMENT customer ANY>
Sequences	<!ELEMENT customer (name,address,company)>
Choices	<!ELEMENT customer (name | company)>
Multiple elements	<!ELEMENT customer (address+, phone*)>
Optional Content	<!ELEMENT customer ((name | company), phone?>

The ATTLIST Rule

- The ATTLIST keyword defines an attribute list
  - The name of the element is followed by the name and type of one or more attributes
- Multiple attributes are defined at the same time
  - Attribute names follow XML naming conventions
- Attribute options indicate whether the user needs to supply the attribute or not
  - #REQUIRED: must be given
  - #IMPLIED: optional, defaults to first value
  - #FIXED: cannot override the value

The ATTLIST Rule

- Attribute values can be ID's and IDREF's
  - Every attribute of type ID must have a different value from each of the others in the document
  - An IDREF attribute contains the value of an ID attribute somewhere in the document

```
<!ELEMENT customer  (address+,  phone*)>
<!ATTLIST Customer
          name          CDATA          #REQUIRED
          sex           (male|female)  #REQUIRED
          custID        ID             #REQUIRED
          companyID     IDREF          #REQUIRED
          isManager     (false|true)   #IMPLIED
>
```

The ENTITY Rule

- Entities represent text to be inserted into the doc
  - They are referenced using the syntax &name;
  - The built in entities are ‘lt’, ‘gt’, ‘amp’, ‘apos’ and ‘quot’
- There are six kinds of entity declaration
  - Character references
  - Parsed/unparsed entities
  - Internal/external entities
  - Parameter entities

ENTITY Declarations

- Character References represent single characters
  - For example ‘&’ is ‘&’ and ‘©’ is ‘©’
- General entities represent strings
  - <!ENTITY copyright “All rights reserved">
- External entities represent content in other docs
  - <!ENTITY copyright SYSTEM “copywright.txt”>
- Unparsed entities represent binary data
  - <!ENTITY copyright SYSTEM “copy1.gif” NDATA GIF>
- Parameter entities are only used inside DTD’s
  - <!ENTITY % commonContent “(title, forename,surname)”>
  - <!ELEMENT customer %commonContent; >